**Amendments to the Claims:**

This listing of claims will replace all prior versions, and listings, of claims in the application:

**Listing of claims:**

1-26. (Cancelled).

27. (New) A transaction queue for an agent operative according to a dynamic priority scheme, the transaction queue operating initially according to a default priority scheme and then engaging a second priority scheme to invalidate selected stored transactions when the transaction queue becomes congested.

28. (New) The transaction queue of claim 27, wherein the second priority scheme invalidates a stored non-posted blind prefetch request.

29. (New) The transaction queue of claim 27, wherein the second priority scheme invalidates a stored non-posted prefetch request that is associated with a posted prefetch request.

30. (New) The transaction queue of claim 27, wherein the second priority scheme invalidates a stored non-posted patterned prefetch request.

31. (New) The transaction queue of claim 27, wherein the agent is a general purpose processor.

32. (New) The transaction queue of claim 27, wherein the agent is a memory controller.

33. (New) A method for managing transactions for an agent, comprising:

receiving a first transaction;

if a transaction queue is congested, identifying a second transaction having a lower priority than the first transaction,

removing the second transaction from the transaction queue, and

storing the first transaction in the transaction queue.

34. (New) The method of claim 33, wherein the transaction queue is congested whenever a latency of stored transactions exceeds a predetermined threshold.

35. (New) The method of claim 33, wherein the transaction queue is congested whenever a number of stored transactions exceeds a predetermined threshold.

36. (New) The method of claim 33, wherein the transaction queue is congested whenever the transaction queue becomes full.

37. (New) The method of claim 33, wherein the first transaction is a core read request and the second transaction is a patterned prefetch request or a blind prefetch request.

38. (New) The method of claim 33, wherein the first transaction is a patterned prefetch request and the second transaction is a blind prefetch request.

39. (New) A transaction queue, comprising:

an arbiter; and

a plurality of queue registers coupled to the arbiter, each queue register to store data representing a stored bus transaction, each queue register having an address field and a status field;

wherein, responsive to a congestion event, the arbiter invalidates a selected stored bus transaction.

40. (New) The transaction queue of claim 39, wherein the congestion event occurs whenever a new bus transaction is received by the arbiter and a number of stored bus transactions exceeds a predetermined threshold.

41. (New) The transaction queue of claim 39, wherein the controller invalidates the selected stored bus transaction by modifying the status field in a queue register corresponding to the selected stored bus transaction.

42. (New) The transaction queue of claim 39, wherein the selected stored bus transaction is a blind prefetch request.

43. (New) The transaction queue of 39, wherein, when there are no valid blind prefetch requests in the transaction queue, the selected stored bus transaction is a patterned prefetch request.

44. (New) A system for managing external bus transactions, comprising:

an agent;

an arbiter coupled to the agent, the arbiter operative to store external bus transactions according to a default priority scheme, the arbiter operative according to a second priority scheme in response to a congestion event;

an internal cache coupled to the arbiter;

a transaction queue coupled to the arbiter, the transaction queue storing the external bus transactions;

an external bus controller coupled to the transaction queue; and

a memory subsystem coupled to the external bus controller through an external bus.

45. (New) The system of claim 44,

wherein the default priority scheme prioritizes core read requests over prefetch requests and the prefetch requests over write requests; and

wherein the second priority scheme prioritizes the core read requests over the write requests and the write requests over the prefetch requests.

46. (New) The system of claim 44, wherein the congestion event occurs whenever a new external bus transaction is received by the arbiter and a number of stored external bus transactions exceeds a predetermined threshold.

47. (New) The system of claim 44, wherein the congestion event occurs whenever a new external bus transaction is received by the arbiter and a latency of stored transactions exceeds a predetermined threshold.

48. (New) The system of claim 44, wherein the agent is a general purpose processor.

49. (New) The system of claim 44, wherein the agent is a memory controller.

50. (New) The system of claim 44, wherein the agent is a prefetch queue.

51. (New) A system, comprising:

a memory system,

an agent coupled to the memory by an external bus, the agent to generate read or write transactions on the bus addressing locations within the memory system, the agent comprising a transaction management system including a queue to store core read requests and prefetch requests, the transaction management system responsive to a congestion event by invalidating non-posted prefetch requests.

52. (New) The system of claim 51, wherein the transaction management system invalidates data of a transaction after data responsive to the transaction is read by the agent.

53. (New) The system of claim 51, wherein the transaction management system comprises a queue having a plurality of entries, each entry to store data of at least two transactions, including a first field representing whether the respective transaction is a core read request or a patterned prefetch request, and a second field representing whether the respective transaction is pending or has been posted on the external bus.

54. (New) A method, comprising:

storing within an agent data representing a plurality of transactions to be conducted external to the agent, the transactions including core read requests, patterned prefetch requests and blind prefetch requests,

storing within the agent data representing a status of the transactions, including whether the respective transactions are pending and whether the respective transaction are in progress externally from the agent, and

responsive to a congestion event, invalidating pending blind prefetch requests that have not been posted externally.

55. (New) The method of claim 54, further comprising, responsive to another congestion event, invalidating pending patterned prefetch requests that have not been posted externally.

56. (New) The method of claim 54, further comprising, posting the transactions externally according to a priority scheme that prioritizes core read requests over patterned prefetch requests and patterned prefetch requests over blind prefetch requests.

57. (New) The method of claim 54, wherein the invalidating comprises changing the status of the pending blind prefetch request to indicate that it has been completed.